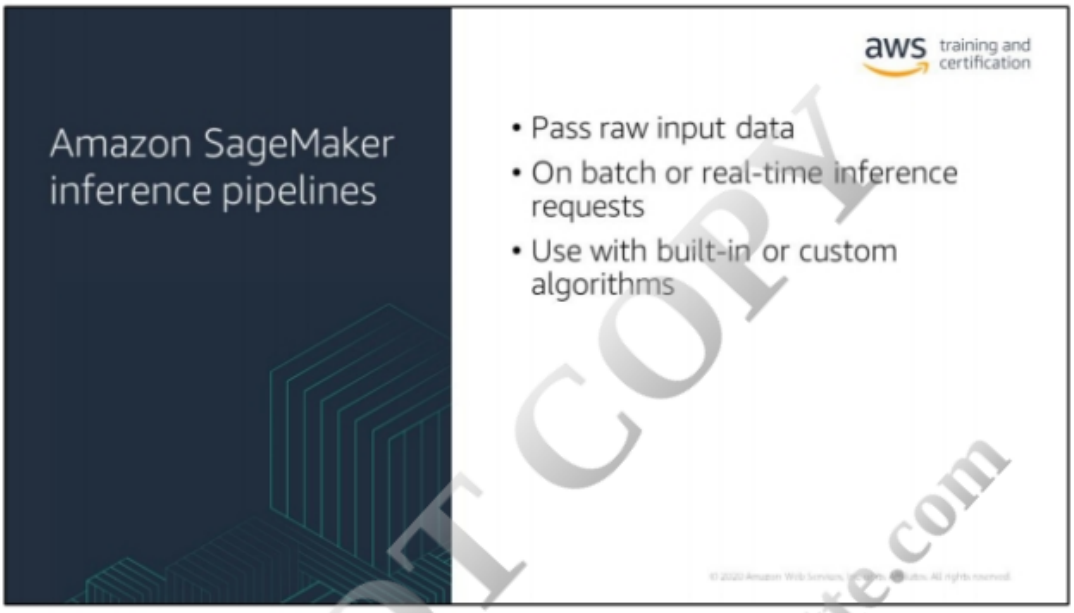
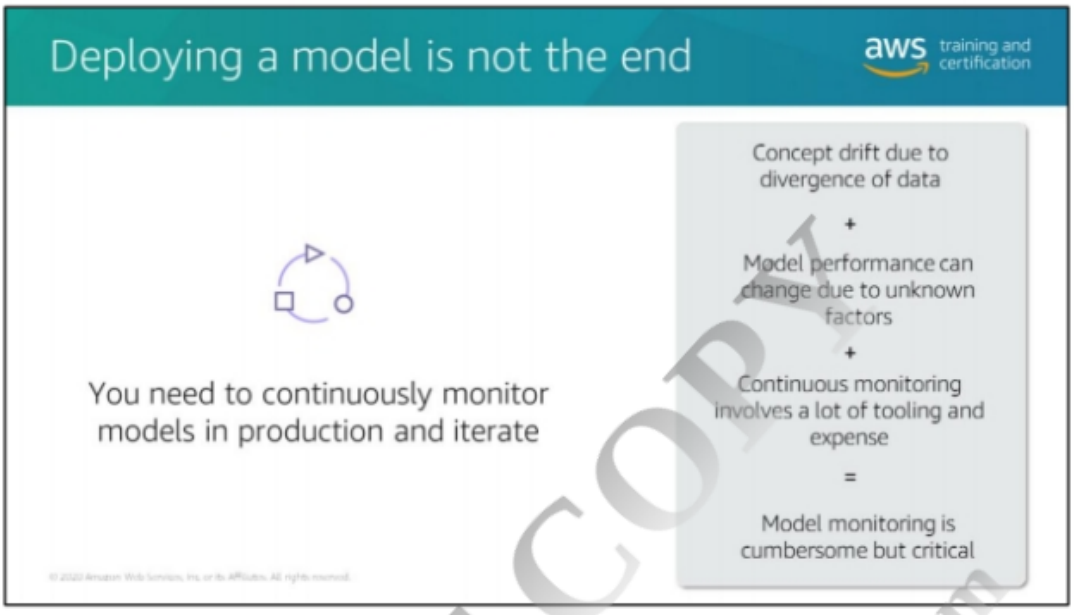
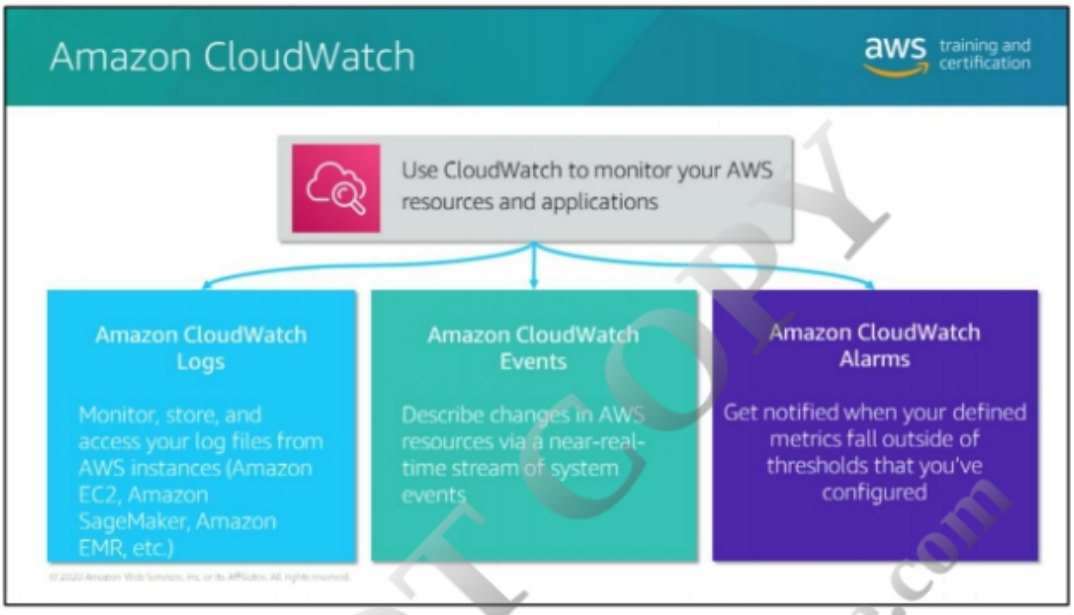


Amazon SageMaker Inference Pipelines enable you to deploy inference pipelines so you can pass *raw* input data, run pre-processing, predictions, and complete post-processing on batch or real-time inference requests. These pipelines are used to define and deploy any combination of pre-trained SageMaker built-in algorithms and your custom ones, packaged in Docker containers. An inference pipeline is a model that is composed of a linear sequence of two to five containers that process requests for inferences on data. You can use these pipelines to combine preprocessing, predictions and post-processing tasks.



Machine learning models are typically trained and evaluated using historical data. But, the real world data may not look like the training data, especially as models age over time and the distributions of data change. The gradual misalignment of the model and real world is known as concept drift, and it can have a big impact on prediction quality.

The way to get around this problem is to continuously monitor the model performance, that is typically done by integrating third-party tools. This can be expensive and may need multiple work-arounds depending on the tool that is chosen. This means it is not scalable and not quick enough to detect quality deviations.



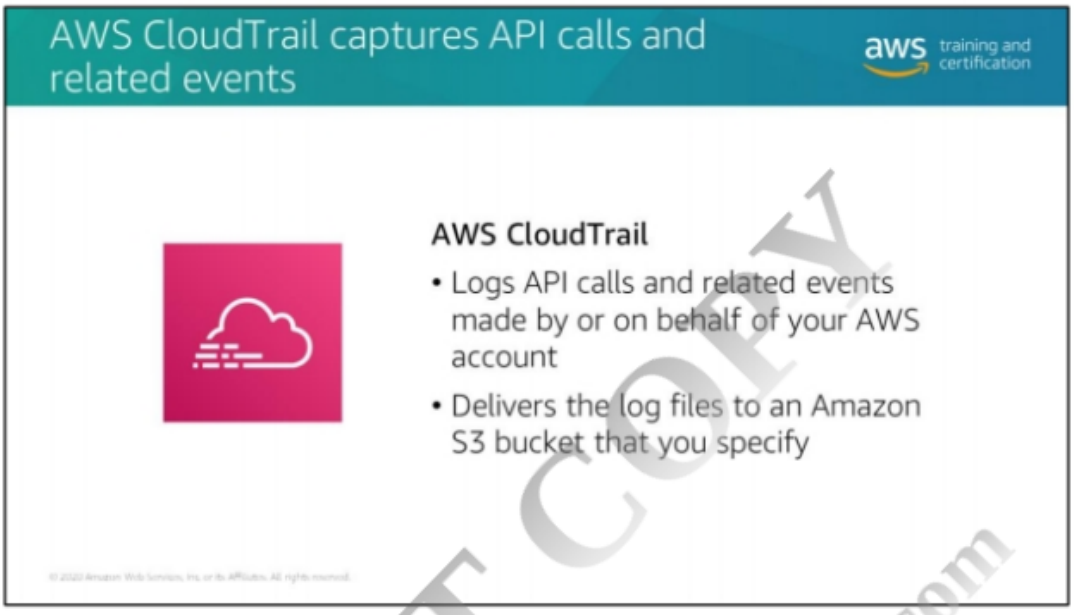
Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon SageMaker and your other AWS solutions. AWS provides monitoring tools to watch Amazon SageMaker, report when something is wrong, and take automatic actions when appropriate. Here's a description of each one.

Amazon CloudWatch monitors your AWS resources and the applications that you run on AWS in real-time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have Amazon CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed.

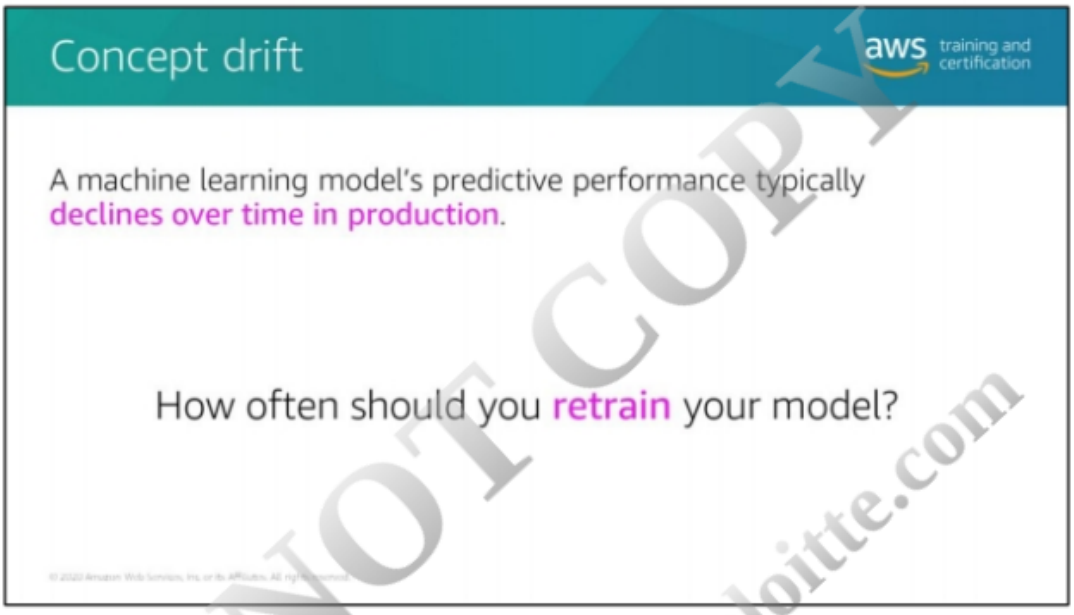
Amazon CloudWatch Logs enables you to monitor, store, and access your log files from Amazon EC2 instances, AWS CloudTrail, and other sources. Amazon CloudWatch Logs can monitor information in the log files and notify you when certain thresholds are met. You can also archive your log data in highly durable storage.

Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. For instance, Amazon CloudWatch Events rules can be created to react to a status change in an Amazon SageMaker training, hyperparameter tuning, or batch transform job.

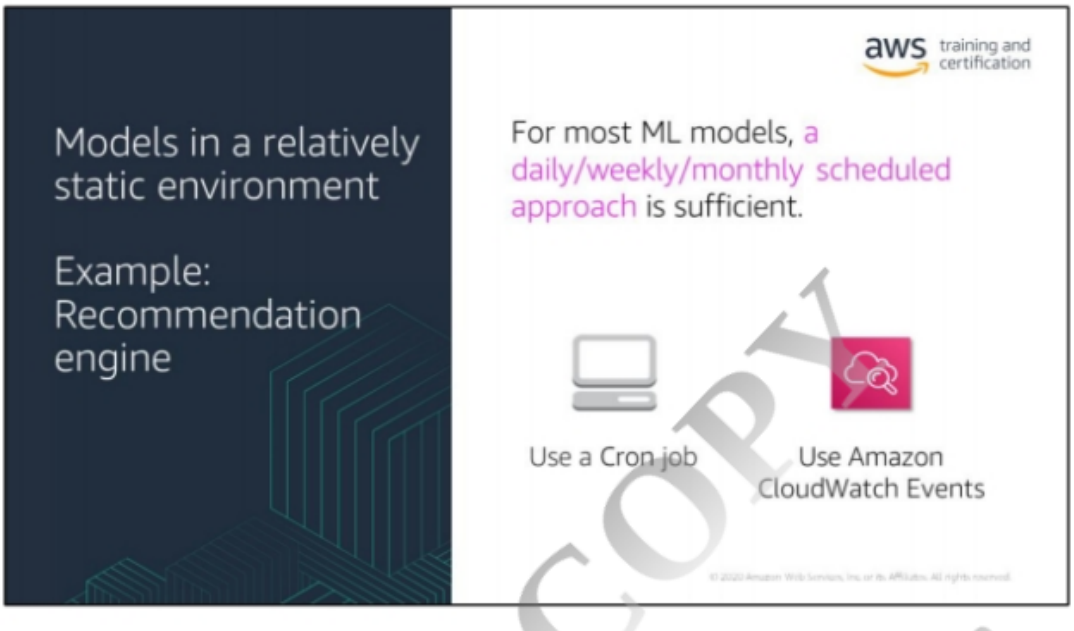
Using Amazon CloudWatch Alarms. You can create a CloudWatch alarm that watches a single CloudWatch metric or the result of a math expression based on CloudWatch metrics. The alarm performs one or more actions based on the value of the metric or expression relative to a threshold over a number of time periods.



AWS CloudTrail captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred.



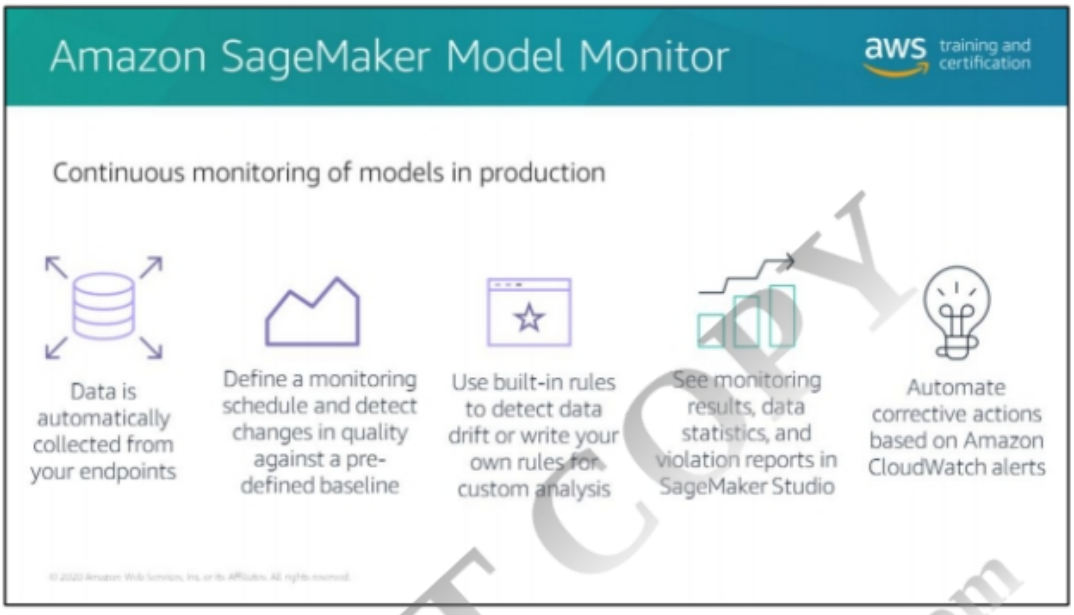
A machine learning model's predictive performance typically declines over time in production. For this example there is an accuracy loss of 1.5%. So the question arises - how often should I retrain my model and what is the best way to retrain my model?



Periodic re-training - For most ML models, daily/weekly/monthly retraining and therefore a simple scheduled approach is sufficient. You can use a cron job or the preferred Amazon CloudWatch Event to trigger a Step Function which then orchestrates the various data prep, training, evaluation and deployment steps.

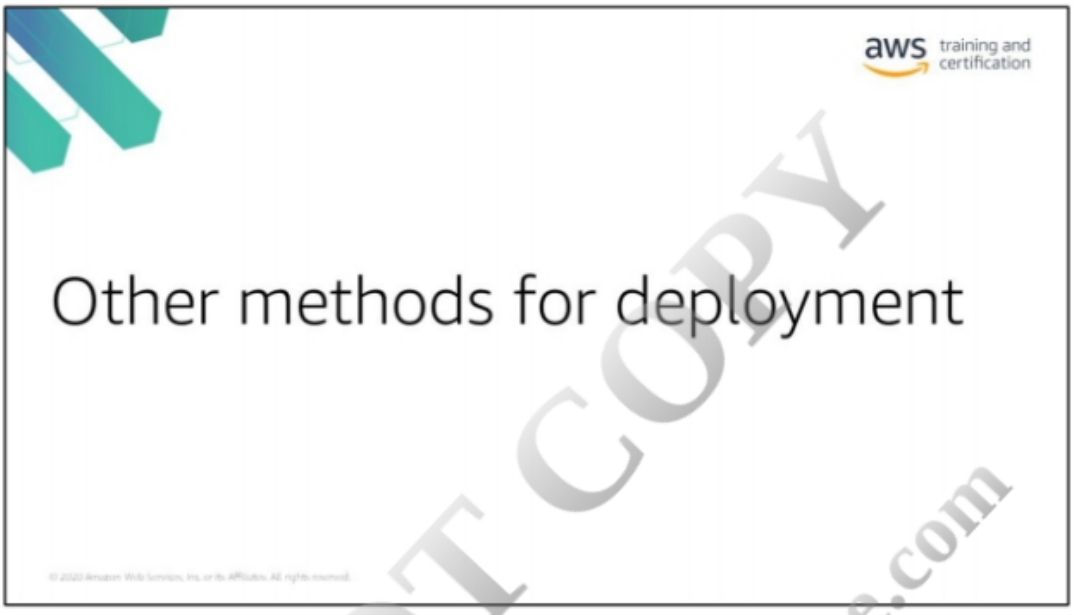
For models that operate in a much more dynamic environment, for example, models that depend on competitive pricing, financial models in the context of changing interest rates; need to be trained more adaptively. In such situations, models should be monitored for changing performance against business metrics and retraining should be automatically triggered if the production model is suddenly under-performing. For example, if your mean prediction is X, and that drops by 10% over a certain time interval, then this may indicate that a sudden concept drift issue has occurred. Then an automatic model retraining process can be triggered via scheduled Lambda/Step Functions.

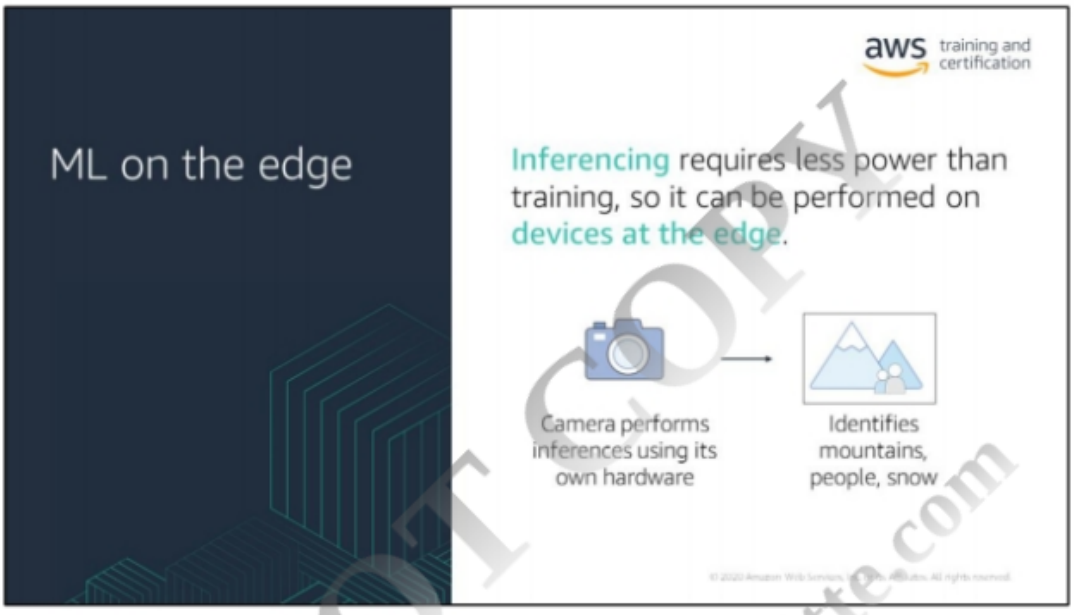
Of course, the retrained models are only effective if they are trained with the new data as it is collected. So, the challenge to consider is the need to collect sufficient new training data from the new "state". Once new model is trained use the above discussed blue/green and A/B deployment strategies to minimize downtime in production.



SageMaker Model Monitor, which is a capability of Amazon SageMaker, emits per-feature metrics to Amazon CloudWatch. These metrics provide alerts when data quality issues appear in production.

- Amazon SageMaker Model Monitor monitors models in production and detects errors so you can take remedial actions.
- Amazon SageMaker Model Monitor eliminates the need to build any tooling to monitor models in production and detect when corrective actions need to be taken.
- It analyzes the data collected based on built-in rules or customer-provided rules at a regular frequency to determine if there are any rule violations. The built-in statistical rules can be used to analyze tabular data and detect common issues such as outliers in prediction data, drift in data distributions compared to training datasets, and changes in prediction accuracy based on observations from the real world.
- With Amazon SageMaker Model Monitor, you can get alerts customers in the Amazon SageMaker Studio interface when rules are violated, and metrics are emitted in CloudWatch so you can set up alarms to audit and retrain models.

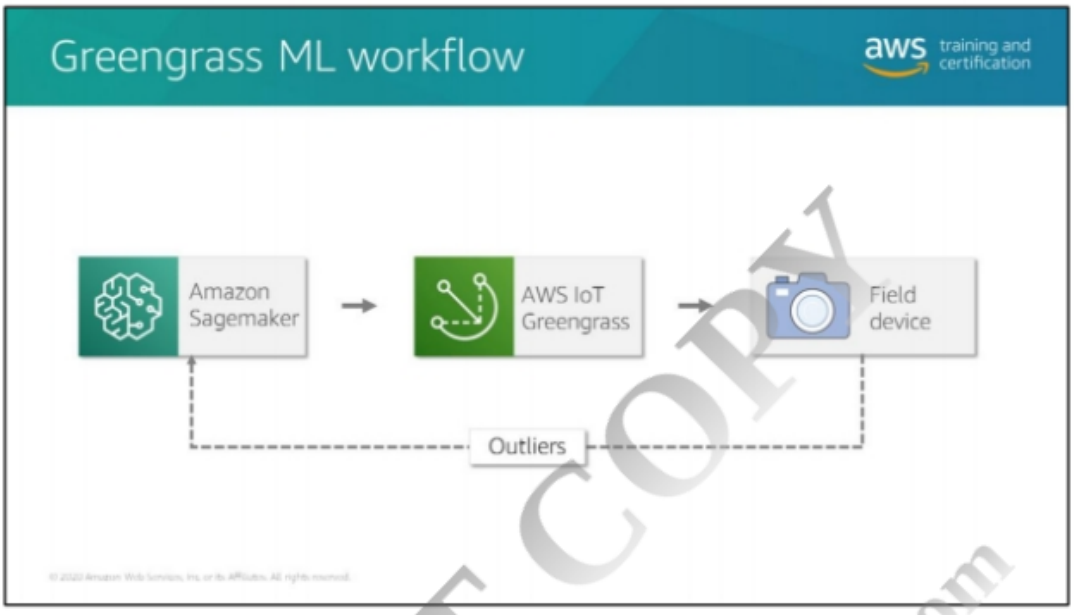




Training your ML models requires the powerful compute infrastructure available in the cloud. However, making inferences against the built models typically requires far less computational power. That means inference is possible on an edge device that has limited power and connectivity. Consider an edge device like a security camera. Instead of having this security camera send the video content up to cloud for analysis, this analysis for identifying unknow people, objects etc. can happen on the camera itself. That is if the camera is equipped with the right machine learning model.

AWS IoT Greengrass enables machine learning on edge devices. AWS IoT Greengrass makes it easy to perform ML inference locally on devices, using models that are created, trained, and optimized in the cloud. ML models built using Amazon SageMaker, AWS Deep Learning AMI, or AWS Deep Learning Containers and persisted in Amazon S3 are deployed on the edge devices.

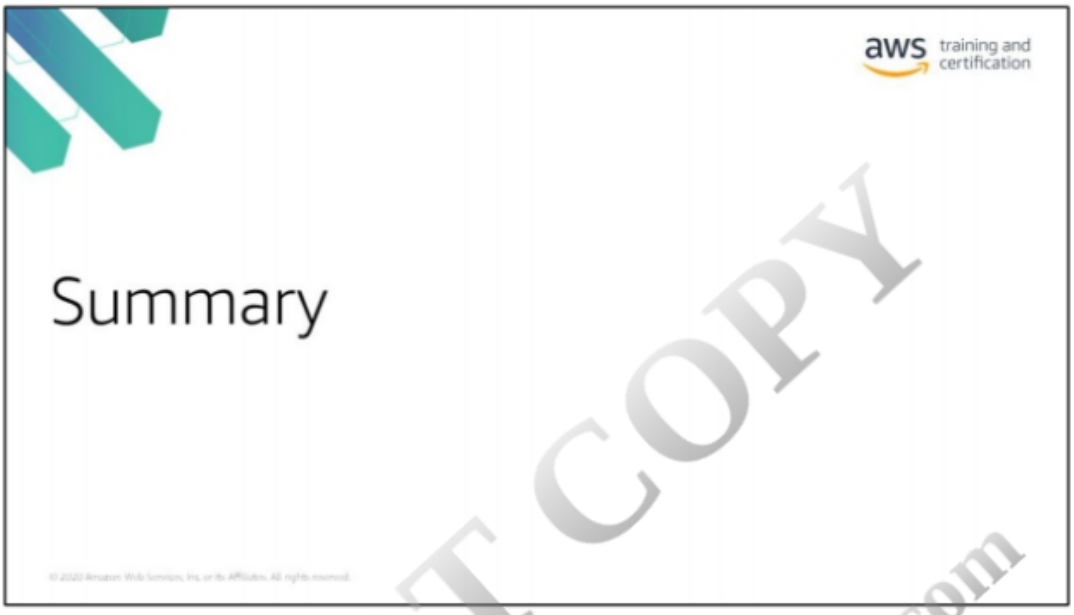
Performing inference locally on connected devices running AWS IoT Greengrass reduces latency and cost. Instead of sending all device data to the cloud to perform ML inference and make a prediction, you can run inference directly on the device. As predictions are made on these edge devices, you can capture the results and analyze them to detect outliers. Analyzed data can then be sent back to Amazon SageMaker in the cloud, where it can be reclassified and tagged to improve the ML model.

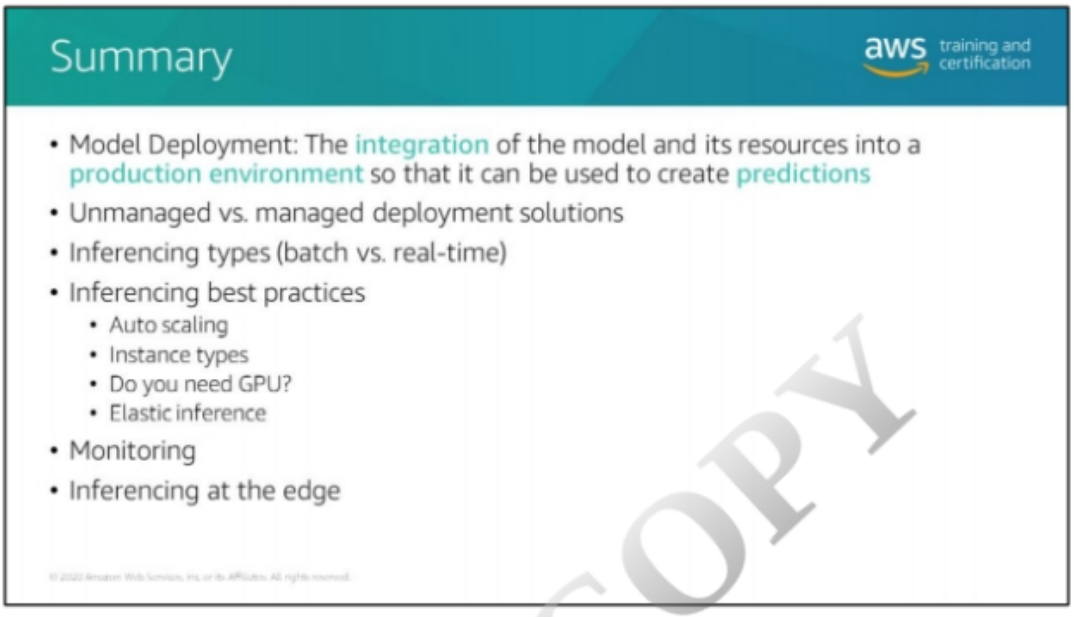


You can use ML models that are built, trained, and optimized in the cloud and run their inference locally on devices. For example, you can build a predictive model in Amazon SageMaker for scene detection analysis, optimize it to run on any camera, and then deploy it to predict suspicious activity and send an alert. Data gathered from the inference running on AWS IoT Greengrass can be sent back to Amazon SageMaker, where it can be tagged and used to continuously improve the quality of the ML models.

Though you could deploy the ML model to multiple platforms, such as Intel or NVIDIA, at the edge and in the cloud, that is not always practical because the ML model is tightly coupled to the framework that you used to train it, such as MXNet, Tensor, or PYTORCH. If you want to deploy an ML model to a platform other than the specific platform that you trained it for, you must first optimize the model. As the number of ML frameworks and platforms increase, the effort required to optimize the models for additional platforms increases and might become prohibitively time consuming.

Amazon SageMaker Neo includes two components to address this problem: a compiler and a runtime. The compiler converts models to an efficient, common format, which is run on the device by a compact runtime that uses less than one-hundredth of the resources that a generic framework traditionally consumes. Amazon SageMaker Neo runtime is optimized for the underlying hardware, and uses specific instruction sets that help speed up ML inference. Models are optimized with less than one tenth of the memory footprint so that they can run on resource constrained devices, such as home security cameras and actuators.





We've covered a lot in this Model Deployment module. Let's take a few minutes to review what we discussed.

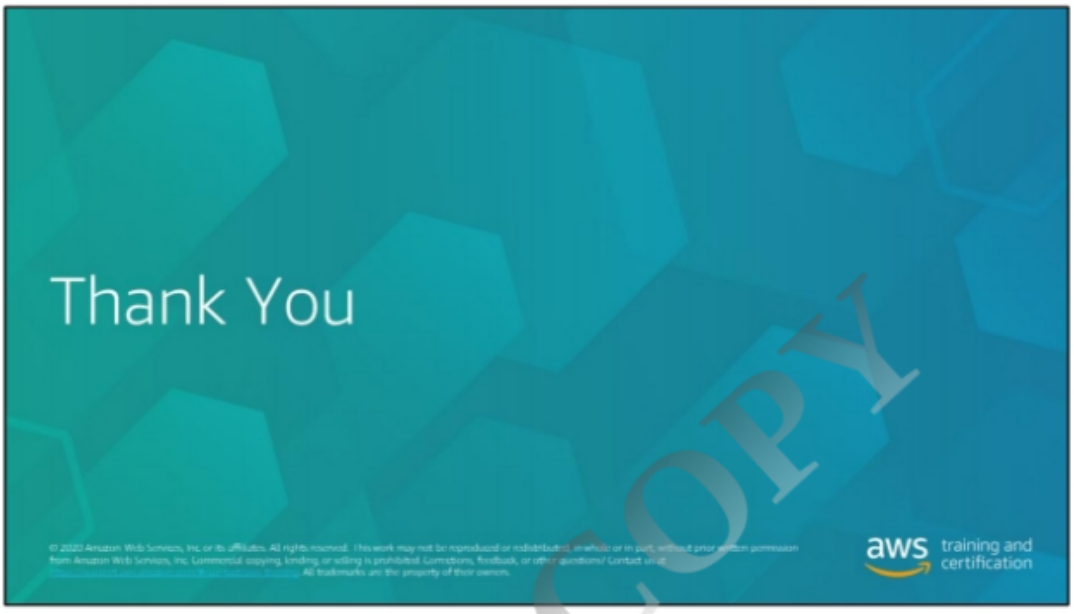
There are still decisions to be made before deploying your model or integrating the model and its resources into a production environment so that it can be used to create predictions.

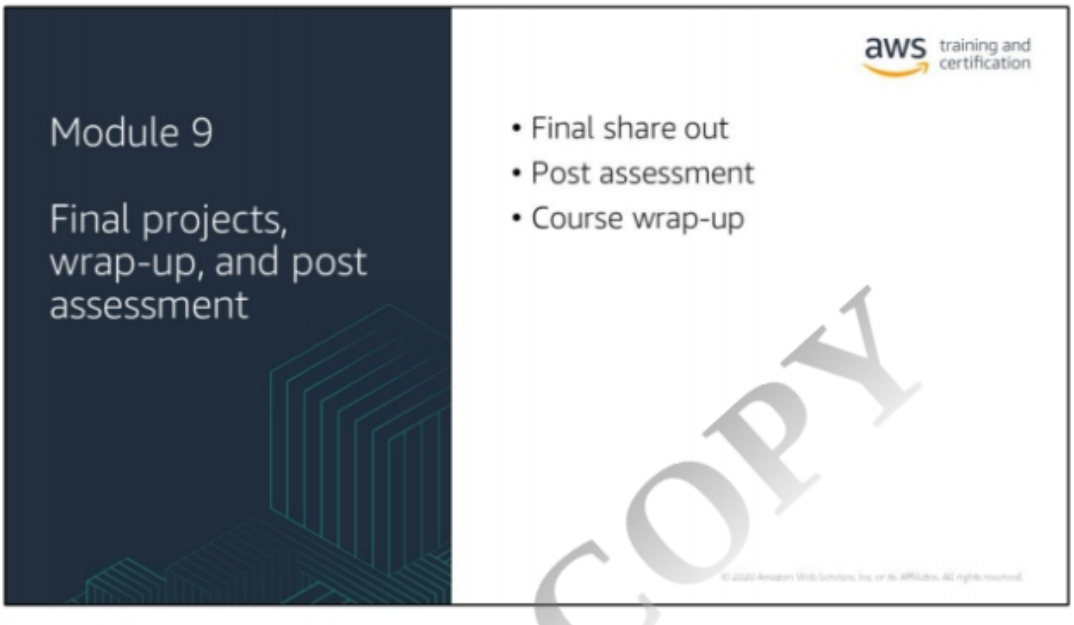
It is possible to host ML models directly on Amazon EC2 instances or containers (like ECS or EKS) and make them available to your consumers. However, if you choose one of those options, you are responsible for creating the AMI (Amazon Machine Image) containing your model artifact, launching one or more EC2 instances with this AMI and configuring the autoscaling options necessary to scale according to the inference traffic patterns. However, on AWS, Amazon SageMaker offers a broad variety of options for deployment and inference, and is the recommended service for deploying (also called hosting) your production ML models. Amazon SageMaker, as you already know is the managed platform for end to end machine learning. It provides model hosting services for model deployment, and provides an HTTPS endpoint where the ML model is available to provide inferences.

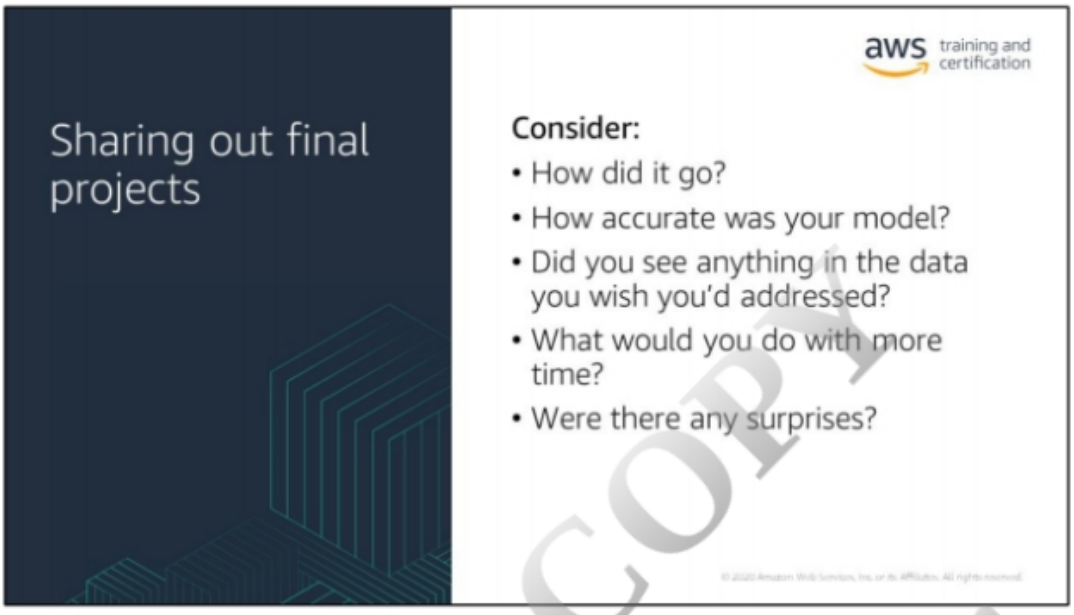
Remember you can use SageMaker to deploy a model to get predictions in 2 ways: You can use Amazon SageMaker batch transform to get predictions for an entire dataset or you can set up a persistent endpoint to get one prediction at a time using Amazon SageMaker Hosting Services.

We also discussed how you need to continuously monitor models in production and iterate them. We talked about how Amazon CloudWatch can be used to monitor your AWS resources and applications and how a machine learning model's predictive performance typically declines over time in production. As a result your model will need to be retrained.

We wrapped up this module talking about how you can use ML models that are built, trained, and optimized in the cloud and yet run the inference locally on devices.







At the end course, students should reflect on their project as a whole:

- How did it go?
- How accurate was your model?
- Did you see anything in the data you wish you'd addressed?
- What would you do with more time?
- Were there any surprises?

Click below for the post-assessment:
https://amazonmr.au1.qualtrics.com/jfe/form/SV_cVj1e29K63ED6mh